ACOUSTIC HANDWRITING RECOGNITION

Oliver M. Perrée
Exeter Mathematics School
Exeter, United Kingdom
oliverperree@gmail.com

March 5, 2020

ABSTRACT

Handwriting recognition using only acoustic data has previously been explored using techniques such as template matching and, recently, deep learning. In this work we investigate ways to create features from audio recordings, and train deep neural network models for performing classification on handwritten digits, and a selection of 72 common English words (48% accuracy; 31% when evaluated on writers whose handwriting wasn't used in training). We also implement a nearest neighbour classifier based on previous works[11][10] using signal difference integral as a similarity measure.

1 Introduction

1.1 Motivations

The idea of acoustic side channel attacks has interested me for a while. The fact that text can be reconstructed from audio recordings of a person typing on a keyboard[14] is concerning from a security perspective, and may be surprising to those to whom it never occurred that this information was present in acoustic emanations in the first place.

Similarly, the sounds produced when a person writes with a pencil on paper carry information about the words which are being written. Handwriting recognition systems that require only acoustic data allow for new and interesting input methods for electronic devices, such as writing numbers on the back of the hand to interact with smartwatches[1].

1.2 Background Information

1.2.1 The handwriting recognition problem

The handwriting recognition problem is typically categorised as either "online" or "offline", depending on the type of data that is given to the handwriting recognition system. Offline handwriting recognition systems receive data in the form of images of text which has already been written. Online handwriting recognition receives a stream of data which is produced as a person is writing – for example, the coordinates and pressure of a stylus detected by a pressure-sensitive tablet. This method provides information beyond that which is available to offline handwriting recognition, such as the speed of the strokes.[2]

Several attempts have been made to recognise handwriting using audio data, recorded using a microphone located near the writing surface [7][11][10]. This is an example of online handwriting recognition.

An acoustic handwriting recognition system could be designed to receive a recording of previously written text, or it could be designed to run in "real time", meaning that it is capable of recognising text as it is being written. In the first case, it does not necessarily need to process the recording quickly, and it may be able to correct incorrectly transcribed words (or whatever unit of text the system operates on) using the context gained from words that come later in the recording. In the second case, the processing time available for the system to identify a word is limited.

A desired property of handwriting recognition systems is the ability to work well for a wide range of handwriting styles. The approaches investigated in this project require a reasonable amount of "training" data, and this aspect of performance is likely to reflect the diversity of the training data.

Unique problems arise when the only information available is audio data. For example, letters and digits which are visually dissimilar might, when written by hand, produce acoustic emissions that are easily confused (for example, the digits 0 and 6).

1.2.2 Audio signal features

While the sequence of samples contains enough information to accurately reconstruct the sound which was recorded (at least the frequencies which are audible to humans), and therefore any acoustic information which may help in recognising handwriting is present in this sequence of samples, it is common to transform this data into formats which are easier to compare meaningfully, or which contain the most relevant information for the task, and less redundancy. This is known as feature extraction.

It is difficult to evaluate the similarity of two signals based on a simple comparison of the raw waves. For example, consider the similarity metric which is the mean value of the absolute difference between each corresponding sample in the signals. If two sinusoidal signals are identical, except for a small phase difference, they will appear to be quite different to each other using this metric.

The Fourier transform is widely used in signal processing. Given a signal, the Fourier transform will determine which frequencies are present. We split a recording into chunks and apply the Fourier to each chunk, giving a 2-dimensional feature showing how the frequencies which are present change over time. This is called a spectrogram.

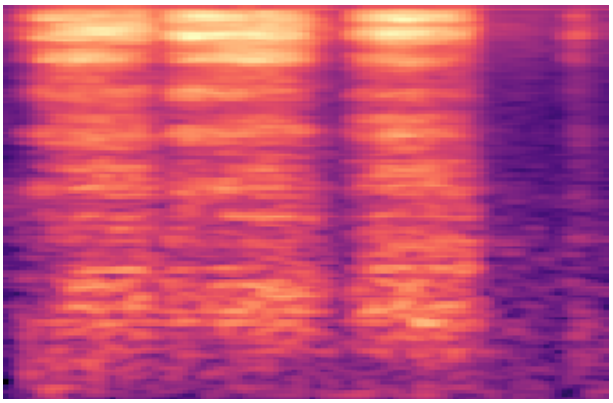

Figure 1: A spectrogram representing the audio of the digit 2 being written.

1.2.3 Template matching

[11] and [10] use a "template matching" approach for acoustic handwriting recognition. This involves processing the training data to construct data representations for each training example, which are then compared to test samples using some similarity measure. A test sample may be predicted to have the same class as the template that it best matches.

This approach might be slow if there are many train and test samples, which must each be compared to a given test sample.

1.2.4 Deep learning

Deep learning systems approximate functions by transforming inputs through a sequence of linear and non-linear functions to produce an output. A standard neural network may compute the data representation at each layer by multiplying the data representation at the previous layer by an adjustable weight matrix. In the supervised learning approach, inputs are passed into the neural network, and the resulting outputs are compared to the desired output. The parameters can then be adjusted using the backpropagation algorithm. This involves computing a gradient of the error with respect to each weight, and subtracting these gradients from the weights, moving the error closer to a minimum. [6]

Deep learning has achieved state-of-the-art results in a wide variety of tasks, including image classification and segmentation, speech recognition and synthesis, natural language processing, playing games, and much more.

1.2.5 Language modelling

In [14], a language model is used for error correction, which was shown to significantly improve performance at recovering words. A spelling checker was used in combination with a trigram language model.

A language model assigns probabilities to sequences of units of text (for example, words or characters). Language models can therefore be used to predict the next word in a sequence, or to predict a missing word given the words surrounding it. The authors of [14] note that incorporating a trigram language model helps the spelling correction system to select the most likely word from the possible correctly-spelled words.

Combining the results of an acoustic handwriting recognition system with a language model has the potential to improve the quality of the reconstructed text.

1.2.6 Security Implications

Acoustic handwriting recognition systems could potentially be misused for the unauthorised reading of sensitive documents as they are being written. While authors of sensitive documents may take care to ensure the documents are not seen by other people, or perhaps even by cameras, they might not ensure that the acoustic emissions of the writing process are not detected by microphones.

2 Related work

2.1 Pen Acoustic Emissions for Text and Gesture Recognition (2009) [11]

Seniuk and Blostein investigated three different similarity measures that can be used to match audio samples to templates created from the training data: the integral of the absolute difference between both signals, the edit distance between the sequence of peaks in both signals, and a measure based on a scale space representation of signals.

2.2 Pen-Chant : Acoustic Emissions of Handwriting and Drawing (2009) [10]

In his masters' thesis Seniuk uses template matching with similar features as in [11]: integral of the absolute difference between smoothed signals, similarity measures between sequences of peaks, and scale-space representations.

2.3 Recognizing Text Through Sound Alone (2011) [7]

Li and Hammond achieve an 80% recognition accuracy on the characters A to Z, using mel-frequency cepstral coefficients and mean amplitude as features. Their approach included endpoint noise removal (removing start and end noises). In a template matching approach, they used dynamic time warping to calculate the distance between queries and templates. Dynamic time warping involves "warping" the signals such that the distance between them is minimised, resulting in a similarity measure which is independent of variations in speed. This is desirable, since there will likely be variations in the speeds at which a person writes the various components of a digit/letter/word.

2.4 Pentelligence: Combining Pen Tip Motion and Writing Sounds for Handwritten Digit Recognition (2018) [9]

Schrapel et al. constructed a pen which emits ink, and records both the motion of the pen and audio, using electronics built into the body of the pen. They found that using motion and sound together resulted in better performance than just using one of the sensors, when using neural networks with majority voting for classification of handwritten digits.

In their conclusion, the authors suggest three applications of their system which could be used "to evaluate the user experience": phone number dialing, a calculator, and password entry.

2.5 WordRecorder: Accurate Acoustic-based Handwriting Recognition Using Deep Learning (2018) [3]

In this paper, Du et al. present a handwriting recognition system that idenntifies words using acoustic emissions from pens and paper. Their method involves segmenting the audio signal into words and letters, producing normalised spectrograms, and using a deep neural network for classification. They also use word suggestion to improve performace. They claim an accuracy of 81% for trained users and 75% for untrained users.

3 Data Collection

3.1 Digits

We collected recordings of people writing individual numerical digits using a pencil. Initially, volunteers were instructed to write digits in boxes in a form printed on A4 paper, pausing for 1 second between each digit, and 3 seconds between each group of 10 digits. A 48 kHz audio recording was made using a OnePlus 3T smartphone, with the bottom of the phone placed on top of the sheet of paper, aligned with an outline on the sheet. This data collection method presented challenges in the data pre-processing phase, in particular the problem of splitting the audio recordings into individual digits and automatically assigning the ground truth labels. Our intention was to split the recordings by periods of near-silence, but this was difficult for a few reasons: volunteers did not pause for exactly the specified time between digits, there are brief periods of near-silence within some digits (such as one way of writing the digit "4"), and the microphone picked up sounds in between the writing of digits (such as the volunteer's hand moving across the paper).

In response to these problems, we developed a new data collection method where volunteers were shown digits on a LibreOffice Impress presentation, and instructed to write each digit anywhere on a sheet of paper using a pencil. The presentation contained 50 digits (each of the digits 0 to 9 five times). Each digit was shown for 2 seconds before the next digit was displayed, and the digits were not displayed in numerical order, so that volunteers could not anticipate the next digit. Unfortunately the timings are not accurate (perhaps due to the time taken to change between slides), but using a short high-pitched tone played as the first digit is displayed, then again after the last digit, we are able to calculate the average time each digit was displayed for, and split the audio recording into chunks of that size. Then, ground truth labels can be matched to the audio segments as the order the digits were displayed in is known. Using this method, audio recordings were made using a OnePlus 3T smartphone. This time, the phone was positioned above the writing surface, with the microphone directed roughly towards the paper. The distance between the phone and the paper was not controlled. Other factors were also uncontrolled - for example, the condition of the pencil, the writing surface (the desk itself, and whether just on a single sheet of paper or a large stack). We recorded a unique writer ID for each recording, so that recordings from the same writer can be identified. Most of the recordings are of the author writing. These are associated with the writer ID `0`.

3.2 Words

We also collected recordings of people writing individual words using a pencil. We selected 75 commonly occurring English words (three of which were later discarded; see Appendix A for the full list), and made recordings using a similar method as with digits, but with a few changes:

- Each word was shown for 3 seconds to give people more time to write them.
- The presentation was converted to a video to ensure accurate slide timings, because it was noticed that there was an unpredictable delay before the high-pitched tone was played at the first digit and after the last digit. This had the advantage that slide timings were now accurate, but we still added both high-pitched tones, to help determine when the sequence of words begins and ends.

This time, the smartphone was typically placed on the same surface as the paper (either directly or on top of another object such as a book), on a nearby surface, or positioned above the writing surface as with the digits recordings. The microphone was always directed roughly towards the paper.

Some recordings were marked as poor quality after they were made (for example, if there was significant background noise).

4 Methodology

4.1 Data preprocessing

4.1.1 Digits

In order to correctly split the recording into individual digits, we need to detect the beeps which mark the beginning and end of the recording. We assume that the first and second beeps happen in the first and last 5 seconds respectively, and use the basic onset detector algorithm from the LibROSA[8] library (`librosa.onset.onset_detect`) to find the times of the beeps. Then we find the average digit duration and split the recording into 50 sections of that duration, corresponding to each digit. These sections are saved as WAV files, and the filenames are associated with the

corresponding label, as well as the writer ID. Then, approximately 20% of the digit examples are randomly chosen and marked as being in the validation dataset.

The recording for the first digit in the sequence, a 7, appears to sometimes contain the initial beep. We didn't check all examples, or attempt to fix the problem, since it was discovered after all the experiments had been run. It is possible that the last digit, which is also a 7, may also be contaminated by the second beep, however 60% of the examples for the digit 7 do not come from the beginning or end and so are not contaminated by the beep.

Mel-scaled spectrograms were produced using `librosa.feature.melspectrogram` and saved as PNG images. `librosa.effects.trim` was used with `top_db=30` in an attempt to trim periods of silence from the beginning and end of each digit before producing the spectrogram.

4.1.2 Words

It was found that the times of the beeps could be determined more accurately by simply taking the absolute value of the wave, normalising the amplitude, and finding the time where a threshold of 50% of the maximum amplitude is exceeded. First, a band-pass filter is applied at the approximate frequency of the beep, then this method is applied to the first 5 seconds of the recording to locate the first beep, and to the last 10 seconds to locate the second beep. For six recordings, this method returned beep times which corresponding with word durations which deviated from the expected 3.00 seconds by more than 0.01 seconds. These recordings were discarded, with no examples from them added to the datasets.

During the data collection process, we noticed that writers often took more than 3 seconds to write the word "business". Therefore, the 3 second window corresponding to the "business" might not contain the full word, and the following 3 second window, which corresponds to the word "who", might contain the end of "business". Also, through listening to some of the audio files, we found that those for the first word, "the", often contained the beep – it would be very easy to classify such examples as being the word "the" simply by checking whether this loud tone of constant frequency is present at the start of the recording. We therefore decided to ignore all examples with classes "business", "who", or "the".

The examples were split into training and validation datasets in three ways. In all three splits, the following conditions hold:

1. Recordings are eligible to be placed in the validation set only if the writer was not the author (writer ID 0).
2. Recordings are eligible to be placed in the validation set only if they are not marked as poor quality.
3. The classes are balanced in both the training and validation sets (the frequency of examples for each class is the same). This is because the splits are made over the recordings, and each recording contains exactly one example for each word.

In the first split, all authors have at least one recording in validation set. For each writer (excluding the author), one recording which is not marked as poor quality is chosen and placed in the validation set. Then, if the validation set contains less than 20% of all recordings, additional recordings to be added to the validation set are randomly chosen from those which are eligible. The resulting split had 21% of recordings in the validation set.

The second split ensures that there is no overlap between the writers who are present in the training set and those who are present in the validation set. The motivation behind this split is to evaluate the performance of systems when they are used for making predictions for writers whose handwriting was not used in training. We produced this split by making multiple candidate splits with at least 20% of the recordings in the validation set, then selecting the split which is closest to 20% (if there are multiple, the first is chosen). Each candidate split is made by randomly choosing a writer (except writer ID 0), and adding all recordings by that writer to the validation set. Then another writer is chosen from those remaining, and so on, until the size of the validation set is greater than or equal to 20%. The result of this process was a split with 21% of recordings in the validation set.

In the third split, all recordings in the training and validation sets are from the same writer. The chosen writer will be the one with the most recordings (excluding the author). The resulting split had 20% of recordings in the validation set.

Mel-scaled spectrograms were produced as with the digits.

4.2 Training deep learning models

We trained convolutional neural networks on mel-scaled spectrograms produced from the audio files. The models used were ResNets[4] and DenseNets[5] of various depths. Out of the six models which were trained on the digits dataset, three were chosen to be run on the words dataset, based on performance and time per epoch.

All models were trained for 100 epochs using a batch size of 32 on a single Nvidia RTX 2070 GPU, using mixed precision. Training was done using the 1cycle policy for adjusting learning rate and momentum, as implemented in the fastai library, with a maximum learning rate of 0.01 (this was found to work well for the digits dataset, and was left unchanged for the words dataset).

The total training time for the 100 epochs ranged from 6 minutes to over an hour depending on which model and which dataset was used.

We performed a grid search over the following boolean hyperparameters:

1. Whether or not the model was pretrained (initialised with weights obtained by training on ImageNet).
2. Whether or not the MixUp[13] augmentation technique was used.

4.3 Building template matching classifier

Inspired by the "template matching" approach in [11] and [10], we implemented what is essentially a nearest neighbour classifier for digits.

We tried using the integral of the absolute difference between the smoothed, normalised amplitudes as the similarity measure, and unsuccessfully attempted to use an edit distance between the sequences of peaks, both based on [10].

The edit distance metric that we attempted to use defined the cost of substituting one peak with another as the difference in time between them, and the cost of inserting or deleting a peak as the height of the peak. We realised after making predictions using this method that the peak heights and times were not normalised: therefore a difference in time of 10 ms corresponded to a cost of 4800, while an insertion or deletion would have a cost of at most 1 (since amplitude was normalised). However, we decided not to rerun this with appropriate costs, due to our inefficient implementation.

We implemented the classifier in Python, using NumPy for operations on arrays.

Signals were passed through a preprocessing function before being used in the nearest neighbour classifier. The preprocessing steps were:

1. Trim silence from the beginning and end of the signal using `librosa.effects.trim` with `top_db=30` (the threshold was chosen somewhat arbitrarily). See Figure 2.
2. Take the absolute value of the signal and apply a Gaussian smoothing filter (with 6 standard deviations corresponding to 10 ms). See Figure 3.
3. Normalise the amplitude of the signal (so the maximum amplitude is 1).
4. Normalise the timescale so that it corresponds to one second.

5 Results and Discussion

Here we present the performance of the different predictive models that were investigated. Full tables can be found in Appendix B.

5.1 Digit classification using convolutional neural networks with spectrogram features

In the following tables, "Pretrained" means the models were pretrained on ImageNet. "Mixup" refers to whether or not the mixup[13] data augmentation technique was used. Time per epoch is measured in minutes and seconds.

When considering the results on the digits dataset, it is necessary to take into account the data quality issues, particularly the fact that some examples for the digit 7 are easily identified by the fact that they contain part of the initial beep (some might also contain part of the final beep).

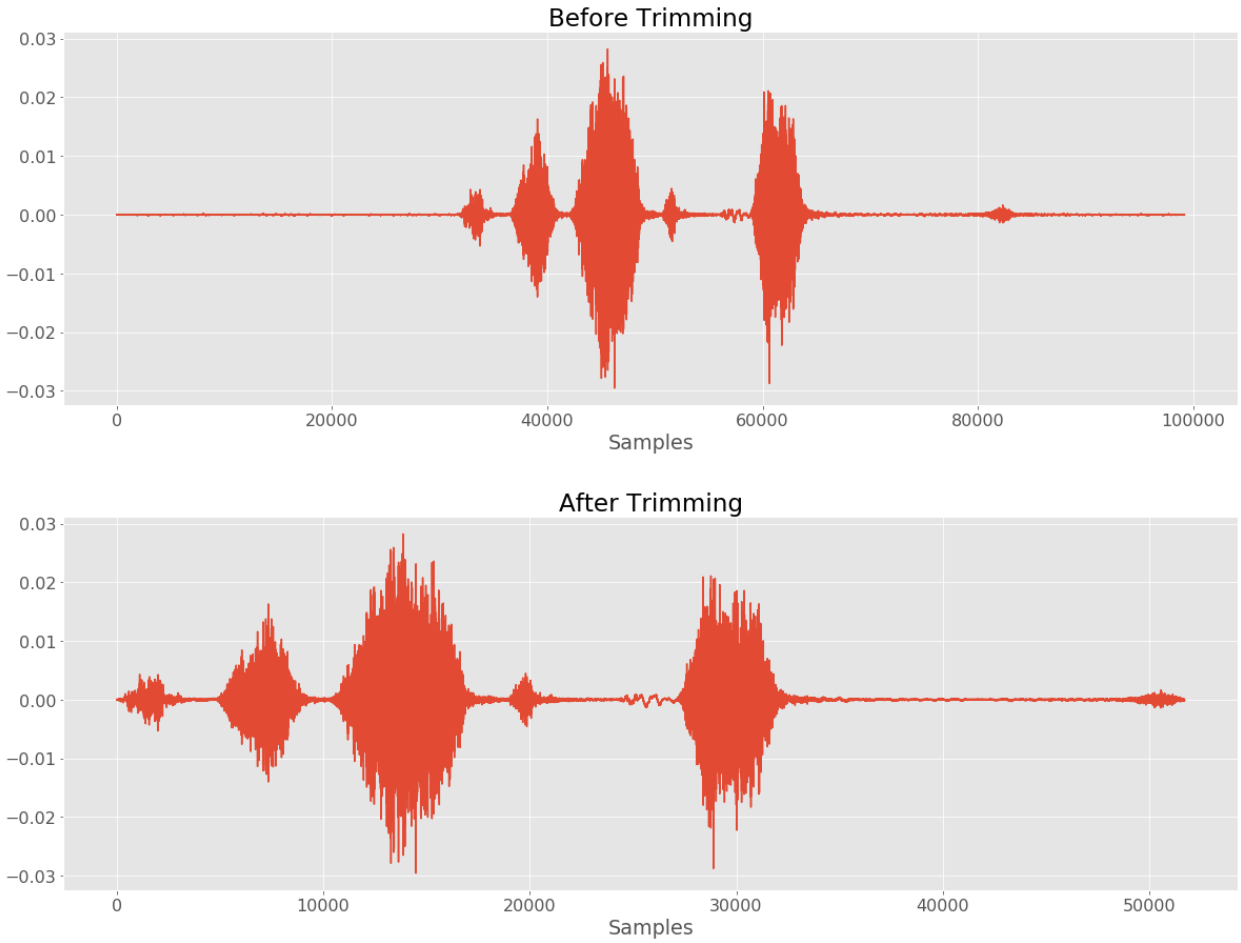

Figure 2: Raw audio signal before and after trimming.

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
resnet34[4]	True	True	71.07	80.36	00:09
resnet50	True	False	63.75	80.54	00:16
densenet121	True	True	76.25	**81.79**	00:19
densenet121	True	False	69.29	80.89	00:19
densenet121	False	False	31.43	80.89	00:24
densenet161	True	True	68.21	80.36	00:34

Table 1: The six best models for digit classification using spectrogram features, measured by accuracy at 100 epochs.

5.2 Digit classification using template matching

The nearest neighbour classifier using the absolute difference integral metric achieved a classification accuracy of 63.57% on the digits dataset and took 47.9 seconds wall time.

We did not manage to get the classifier using signal peak edit distance to perform accurately, or efficiently. The edit distance algorithm, which we implemented in Python and ran using a single thread, took approximately 60 ms to compare two signals, which corresponded to taking over 16 hours wall time to make predictions on the validation set. The accuracy achieved with this similarity metric was 13.75%, only slightly better than the expected accuracy of randomly guessing the class. We suspect that this may be due to the imbalanced costs associated with insertions/deletions and substitution, but could equally be due to a bug in the implementation.

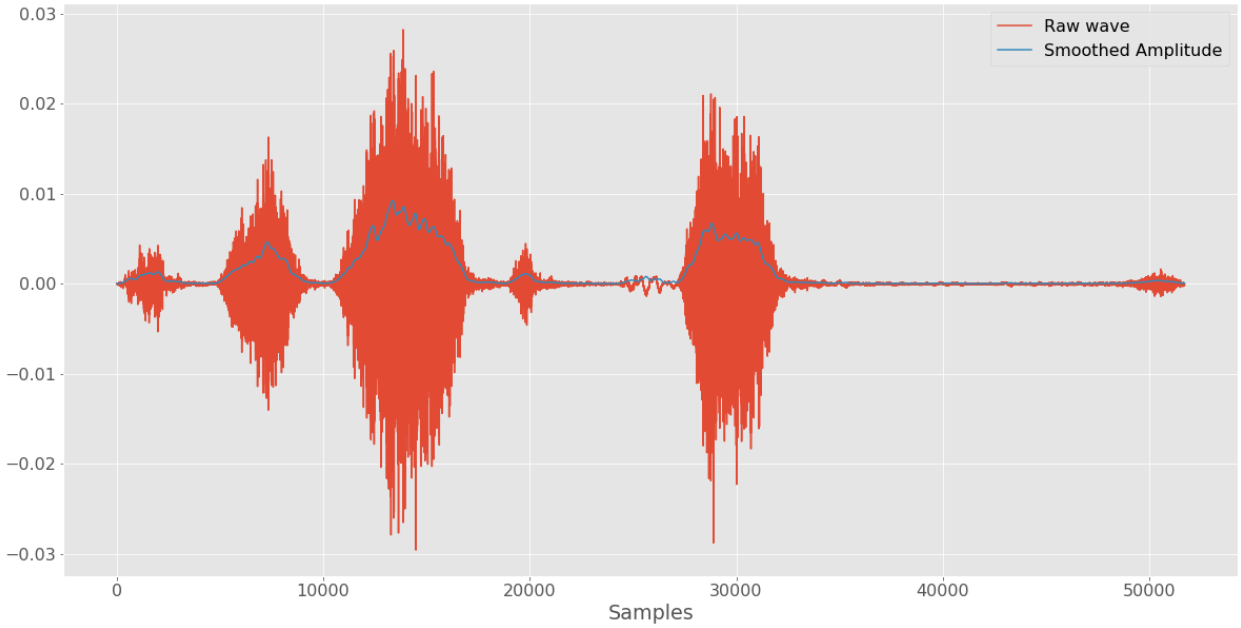

Figure 3: Raw wave and corresponding smoothed amplitude signal.

5.3 Word classification using convolutional neural networks with spectrogram features

5.3.1 Split 1 (all writers except the author have at least one recording in validation set)

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
densenet121	True	True	25.28	**48.75**	00:25
densenet121	True	False	17.36	45.69	00:25
densenet121	False	False	21.53	42.50	00:32
densenet161	True	True	27.78	46.53	00:46
densenet161	True	False	24.44	47.36	00:46

Table 2: The five best models for word classification using spectrogram features, on split 1, measured by accuracy at 100 epochs.

5.3.2 Split 2 (validation set contains different writers to training set)

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
densenet121	True	True	13.38	**31.31**	00:25
densenet121	True	False	10.35	26.01	00:25
densenet121	False	True	6.94	25.38	00:32
densenet161	True	True	15.28	26.26	00:45
densenet161	True	False	12.75	26.77	00:45

Table 3: The five best models for word classification using spectrogram features, on split 2, measured by accuracy at 100 epochs.

5.3.3 Split 3 (all examples come from one writer)

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
densenet121	True	False	14.93	**43.06**	00:09
densenet121	False	True	2.08	40.28	00:11
densenet161	True	False	8.33	39.58	00:15
densenet161	False	True	1.74	37.85	00:19
densenet161	False	False	9.38	40.97	00:19

Table 4: The five best models for word classification using spectrogram features, on split 3, measured by accuracy at 100 epochs.

6 Conclusion

The results demonstrate that the acoustic emanations caused by handwriting contain information about the text that is being written, and suggest that it may be possible to a moderate degree of accuracy.

Potential areas for future research:

- Attempt to recognise full sentences, possibly by performing classification at the word level.
- Combine results of word classification models with language models in order to improve accuracy.
- Use signals from multiple microphones.

References

[1] Chen et al. "WritePad: Consecutive Number Writing on Your Hand With Smart Acoustic Sensing". In: IEEE Access (2018).

[2] Dalbir and Singh. "Review of Online & Offline Character Recognition". In: International Journal Of Engineering And Computer Science 4 (May 2015), pp. 11729–11732. ISSN: 2319-7242.

[3] Haishi Du et al. "Wordrecorder: Accurate acoustic-based handwriting recognition using deep learning". In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE. 2018, pp. 1448–1456.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 770–778.

[5] Gao Huang et al. "Densely connected convolutional networks". In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, pp. 4700–4708.

[6] LeCun, Bengio, and Hinton. "Deep learning". In: nature 521.7553 (2015), p. 436.

[7] Li and Hammond. "Recognizing Text Through Sound Alone". In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. 2011.

[8] Brian McFee et al. librosa/librosa: 0.7.1. Version 0.7.1. Oct. 2019. DOI: 10.5281/zenodo.3478579. URL: https://doi.org/10.5281/zenodo.3478579.

[9] Maximilian Schrapel, Max-Ludwig Stadler, and Michael Rohs. "Pentelligence: Combining Pen Tip Motion and Writing Sounds for Handwritten Digit Recognition". In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018, pp. 1–11.

[10] Seniuk. "Pen-Chant : Acoustic Emissions of Handwriting and Drawing". MA thesis. School of Computing, Queen's University, Sept. 2009.

[11] Seniuk and Blostein. "Pen Acoustic Emissions for Text and Gesture Recognition". In: 2009 10th International Conference on Document Analysis and Recognition. 2009.

[12] Rachael Tatman. English Word Frequency. Version 1. URL: https://www.kaggle.com/rtatman/english-word-frequency.

[13] Hongyi Zhang et al. "mixup: Beyond empirical risk minimization". In: arXiv preprint arXiv:1710.09412 (2017).

[14] Zhuang, Zhou, and Tygar. "Keyboard Acoustic Emanations Revisited". In: Computer and Communications Security, 12th ACM Conference on. ACM. 2005, pp. 373–382.

Appendices

Appendix A List of words classes

We used the following words in our word classification dataset (crossed out words were recorded but discarded in preprocessing):

- ~~the~~
- of
- and
- to
- a
- in
- for
- is
- on
- that
- by
- this
- with
- I
- you
- it
- not
- or
- be
- are
- from
- at
- your
- as
- have
- all
- new
- more
- an
- was
- we
- will
- home
- can
- us
- about
- if
- page
- my
- has
- search
- free
- but
- our
- one
- other
- do
- service
- no
- time
- they
- site
- he
- up
- may
- what
- which
- their
- news
- out
- use
- any
- there
- see
- only
- so
- his
- when
- contact
- here
- ~~business~~
- ~~who~~
- web
- also
- now

These words were manually selected to be an appropriate length, and are among the most common English words on the Web, according to [12].

Appendix B Results tables for all models

B.1 Digit classification using convolutional neural networks with spectrogram features

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
resnet18	True	True	69.11	77.50	00:06
resnet18	True	False	65.00	75.00	00:06
resnet18	False	True	71.43	76.43	00:08
resnet18	False	False	71.79	78.21	00:08
resnet34	True	True	71.07	80.36	00:09
resnet34	True	False	64.29	77.68	00:09
resnet34	False	True	61.25	76.25	00:12
resnet34	False	False	68.04	78.75	00:12
resnet50	True	True	71.43	78.04	00:16
resnet50	True	False	63.75	80.54	00:16
resnet50	False	True	42.14	77.68	00:21
resnet50	False	False	60.71	76.25	00:19
resnet101	True	True	69.11	80.00	00:24
resnet101	True	False	73.04	79.29	00:24
resnet101	False	True	47.50	74.29	00:30
resnet101	False	False	59.11	76.61	00:30
densenet121	True	True	76.25	81.79	00:19
densenet121	True	False	69.29	80.89	00:19
densenet121	False	True	68.75	78.57	00:24
densenet121	False	False	31.43	80.89	00:24
densenet161	True	True	68.21	80.36	00:34
densenet161	True	False	68.57	80.00	00:34
densenet161	False	True	68.04	77.86	00:44
densenet161	False	False	56.43	80.00	00:44

Table 5: Digits

B.2 Word classification using convolutional neural networks with spectrogram features

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
resnet50	True	True	20.00	39.31	00:20
resnet50	True	False	17.92	36.39	00:20
resnet50	False	True	3.75	26.53	00:25
resnet50	False	False	11.39	30.14	00:25
densenet121	True	True	25.28	48.75	00:25
densenet121	True	False	17.36	45.69	00:25
densenet121	False	True	13.33	40.00	00:32
densenet121	False	False	21.53	42.50	00:32
densenet161	True	True	27.78	46.53	00:46
densenet161	True	False	24.44	47.36	00:46
densenet161	False	True	8.75	33.75	00:58
densenet161	False	False	14.44	40.69	00:58

Table 6: Split 1

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
resnet50	True	True	14.27	23.48	00:20
resnet50	True	False	8.59	20.96	00:20
resnet50	False	True	2.78	14.39	00:25
resnet50	False	False	6.69	18.81	00:25
densenet121	True	True	13.38	31.31	00:25
densenet121	True	False	10.35	26.01	00:25
densenet121	False	True	6.94	25.38	00:32
densenet121	False	False	11.99	23.11	00:31
densenet161	True	True	15.28	26.26	00:45
densenet161	True	False	12.75	26.77	00:45
densenet161	False	True	10.86	23.11	00:57
densenet161	False	False	9.22	22.10	00:57

Table 7: Split 2

Model	Pretrained	Mixup	Accuracy at 20 epochs	Accuracy at 100 epochs	Time per epoch
resnet50	True	True	11.81	29.17	00:07
resnet50	True	False	10.07	34.38	00:07
resnet50	False	True	3.12	26.39	00:09
resnet50	False	False	2.08	27.43	00:09
densenet121	True	True	13.89	35.42	00:09
densenet121	True	False	14.93	43.06	00:09
densenet121	False	True	2.08	40.28	00:11
densenet121	False	False	7.99	37.15	00:11
densenet161	True	True	14.58	31.60	00:15
densenet161	True	False	8.33	39.58	00:15
densenet161	False	True	1.74	37.85	00:19
densenet161	False	False	9.38	40.97	00:19

Table 8: Split 3